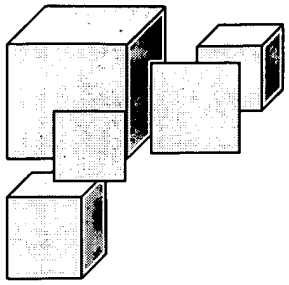
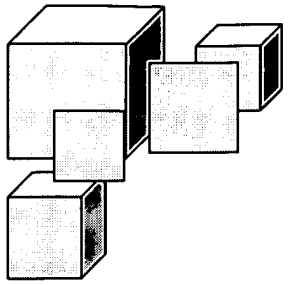
EXHIBIT 3



SDC Architecture

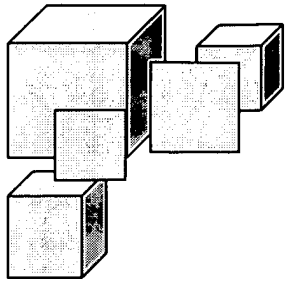
SANRAD
Intelligent Storage Networks



Introduction

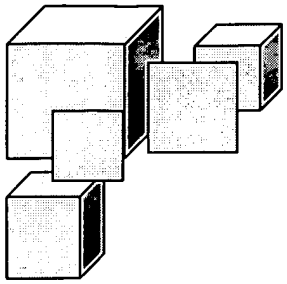
Goal:

- Learn about the SDC architecture and data flow.
- Better understanding of global system functionality.
- Help integration and future development.
- Get suggestions and comments for improvement.



Agenda

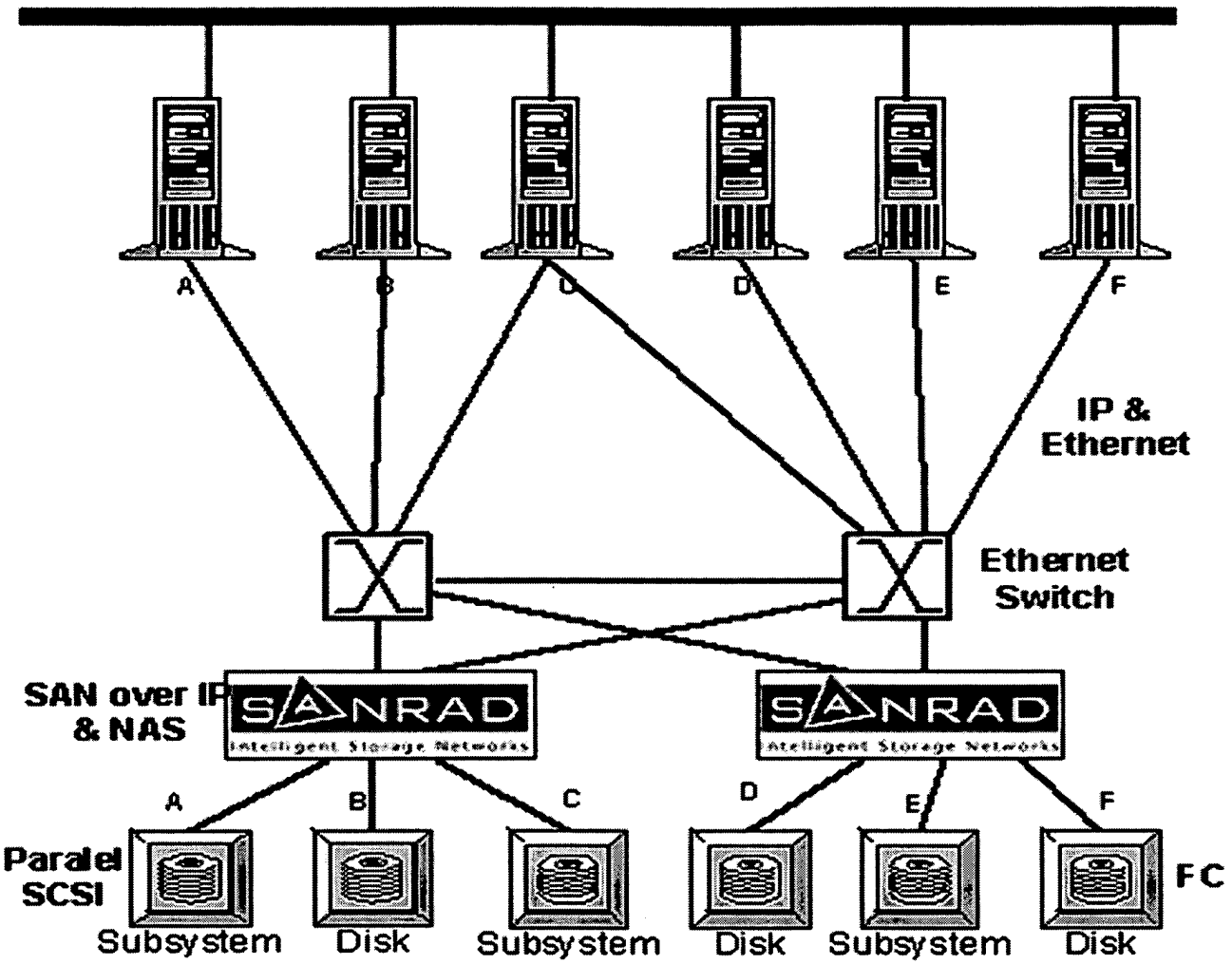
- SDC in its environment - general data flow
- Software modules – general roles and interaction
- Data flow between modules
  - System init
  - Cmd processing
- System Flow Control
- System Data Buffers
- Software layout on hardware and data flow



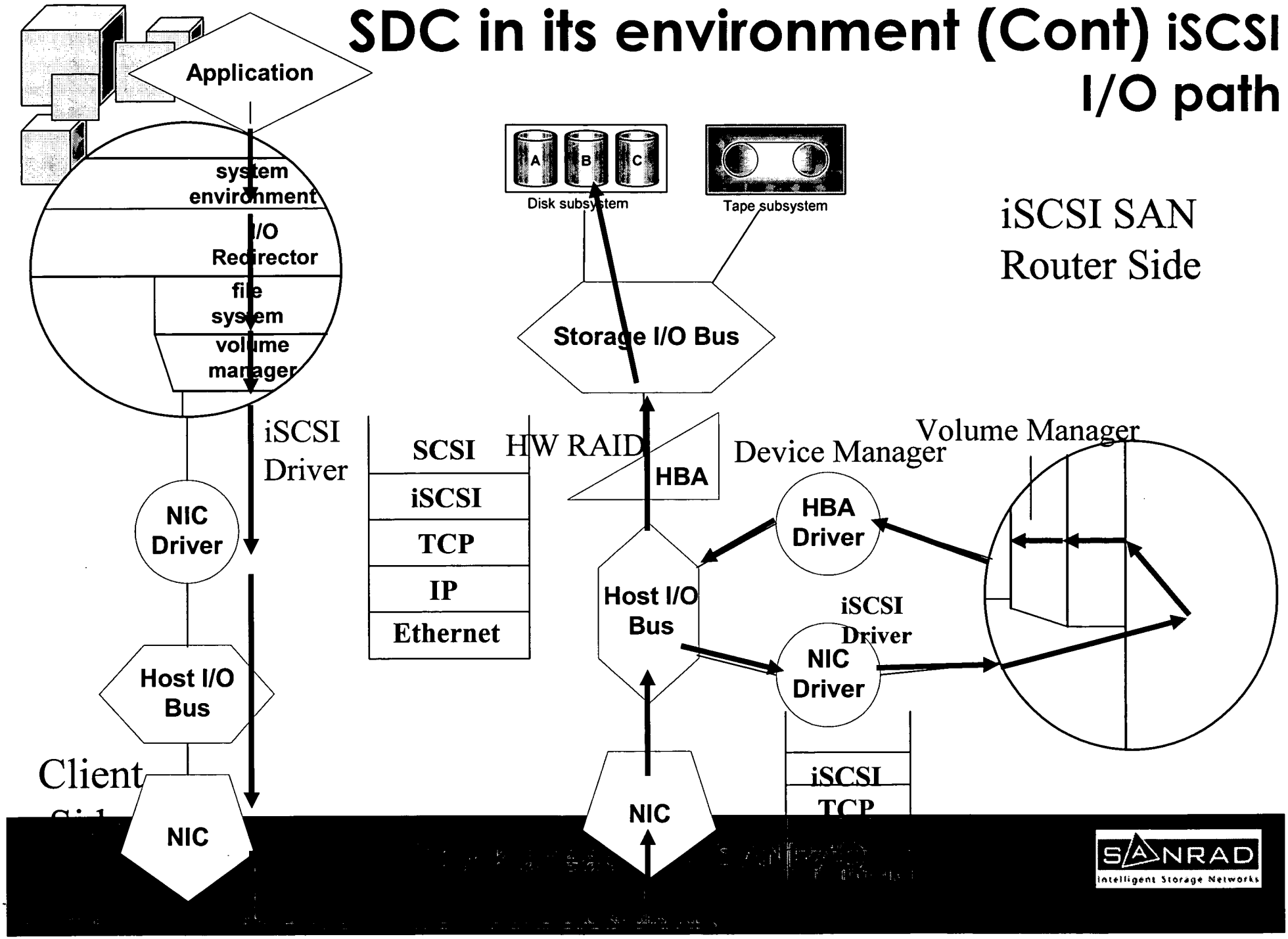
Vocabulary

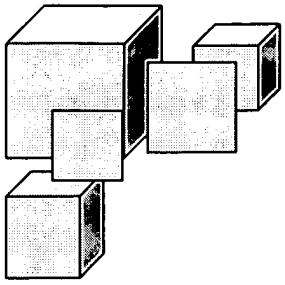
- SDC – Sanrad (was Storage) Domain Controller
- iSCSI – internet SCSI
- PDU – Protocol Data Unit
- FC – Fibre Channel
- P-SCSI – Parallel SCSI
- GE – Gigabit Ethernet
- MAC – Media Access Control

SDC in its environment



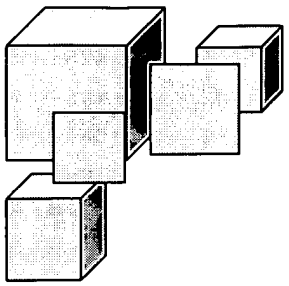
SDC in its environment (Cont) iSCSI I/O path





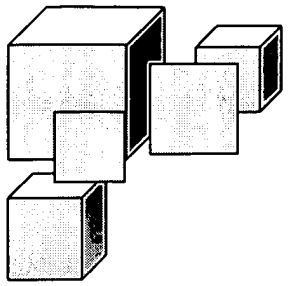
SDC in its environment (Cont)

- The correct perception of a SCSI command
  - A command consists of more than one transfer i.e. multiple frames are exchanged per command. This is true for iSCSI as well as for FC (and P-SCSI??).
- Main difference between target and initiator
  - The Target decides when and how data is transferred.
  - A "normal" initiator views SCSI as a function call. Much like RPC.
  - A target may interface to the SCSI transport layer i.e. the target is aware of the continuation of a command.



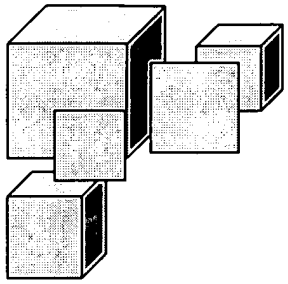
SDC in its environment (Cont)

- General data flow, command's life cycle
  - The SDC is actually a data forwarding device.
  - It acts as a SCSI proxy since it terminates SCSI sessions.
  - The lifetime of a SCSI command is composed of two steps:
    - Decision making: where the command should go to or come from.
    - Data transfer.



SDC in its environment (Cont)

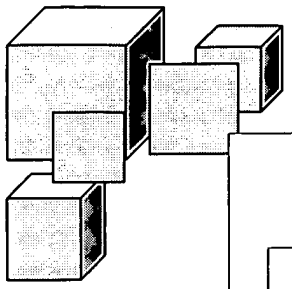
- Main difference between how we implement FC/P-SCSI and iSCSI
  - FC / P-SCSI is implemented in a separate CPU → Main CPU is not bothered with SCSI command until it is over → Minimum interrupts → Good performance.
  - iSCSI is fully implemented by us. iSCSI sits on top of TCP/IP → Interrupt per packet (worse case) → Heavy CPU load → Performance ??
  - In FC / P-SCSI we are not (fully) aware of the "session" bring-up and teardown.
  - iSCSI is fully aware of its sessions.

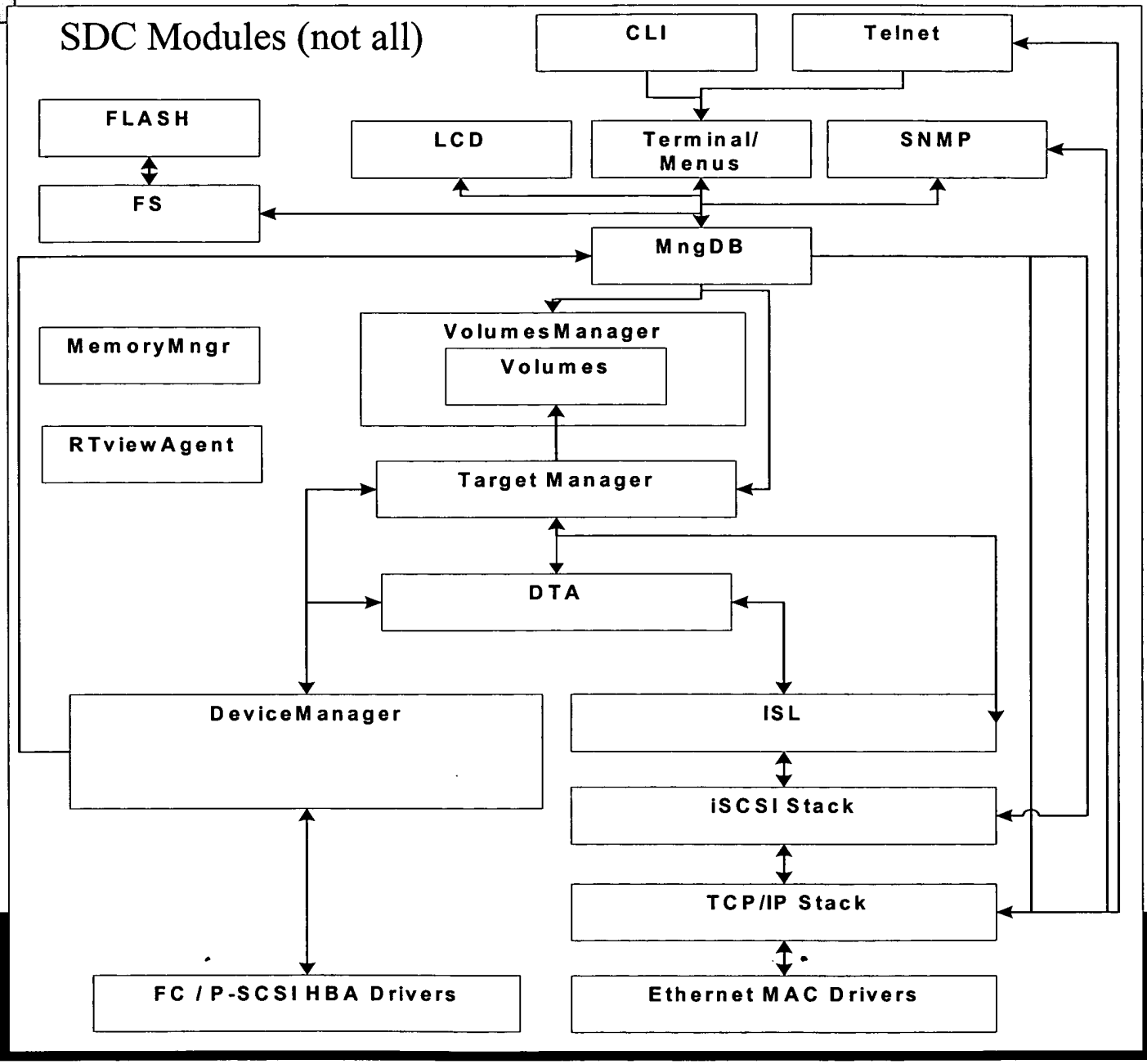


SDC in its environment (Cont)

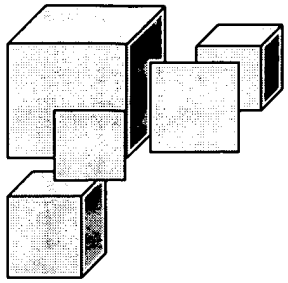
- Side effect of our FC/P-SCSI implementation
  - The SDC will not be transparent. That is, a single SCSI command, from iSCSI, might be split to multiple SCSI commands, to FC.
  - This forces us to have knowledge of every SCSI command/device that we want to support (virtually).
  - Problems with implementing pass through (trnasparent target/LU).
  - Command latency and limited number of commands.

Software modules



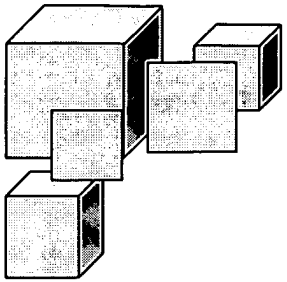


SANRAD
Intelligent Storage Networks



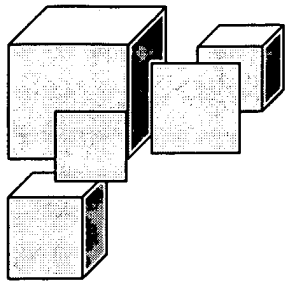
Software modules

- CFE – Common Firmware Environment
  - System Booter
  - Initialize hardware
  - Run power-up tests
  - Load VxWorks from network
    - For development
  - Load VxWorks from flash
    - For production
  - Upgrade SDC software
    - Upgrade to a new application version. In the future via management.
    - Upgrade CFE
  - Resides on SDC flash



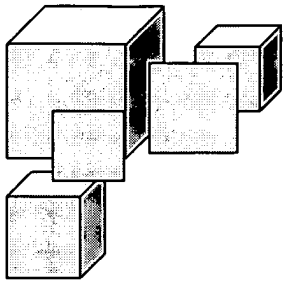
Software modules

- EMM – Embedded Management Module
  - Includes all management components of the SDC
  - System initialization
    - Initializes all system modules
    - Load system configuration from flash and configure all modules with information
  - Supply database functionality
    - System configuration is saved in the database and in non-volatile storage (flash).
    - Database acts as a central event dispatching element for cross module updates. For example:
      - User configuration changes
      - Target/LU disappeared
  - CLI – Command Line Interface
    - Via serial management port
    - Via Telnet
  - SNMP Agent – for management via SNMP



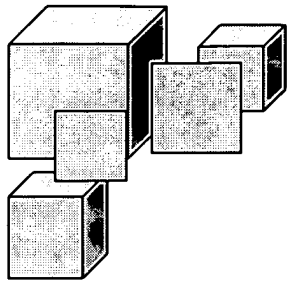
Software modules

- Volumes/VolumeManager
  - The heart of the virtualization block mapping
  - A consulting module that takes as input one logical command and return a list of physical commands.
  - A top level volume has a 1-1 relationship with an LU that the SDC exposes.
  - A volume itself may be composed out of other volumes
    - To achieve concatenation, striping, mirroring.
    - To achieve multi level volumes (e.g. mirror over stripe).
  - VolumeManager is responsible for:
    - Constructing physical volumes and tying them to their remote LU (!!).
    - Constructing volume trees to form multi level volumes.
    - Interacts with EMM to build the volumes.



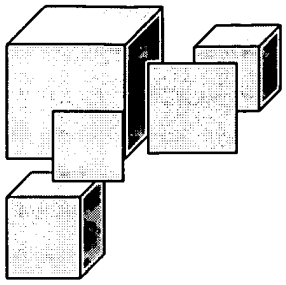
Software modules

- TargetManager
  - This module implements the SCSI level logic in the SDC.
    - The only location in the system that CDBs are parsed.
  - Follows SAM (SCSI Architecture Model) guidelines:
    - Targets with their LUs, Target Port
    - Target Port with its Task Router
    - LU with their Task Sets, Device Server, …
  - Interacts with EMM to build targets and their LUs that the SDC exposes.
    - Tying the LUs to their Volumes
  - Interacts with the SDC's target side (ISL) to:
    - Configure it to accept initiators for new targets.
    - Accepting new initiators and building their I_T_Nexus, I_T_L_Nexus.
    - Accepting SCSI commands form these initiators.



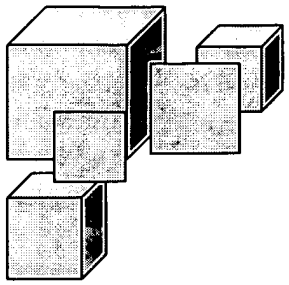
Software modules

- TargetManager (Cont)
  - Interacts with the SDC's initiator side (DeviceManger) to:
    - Implement SCSI commands other than simple read and writes.
  - Consults with Volumes in order to translate a logical command to its physical representation.
  - Schedules commands to be executed and spawns DTAs to carry out the SCSI commands.
  - Receives completion notifications from DTA for forward progress of the LU's Task Set.



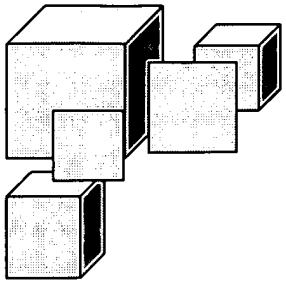
Software modules

- DTA – Data Transfer Arbiter
  - Performs the actual data transfer between SDC's Target side and SDC's Initiator side.
  - Responsible for executing the translation of the logical command to the physical commands.
  - For each SCSI Command exists a DTA object that is responsible for its execution.
  - Receives the logical command and the physical command list from the TargetManager (actually from the DeviceServer of the LU that the logical command belongs to).
  - Uses both ISL and the DeviceManager for passing the data between initiator and target.



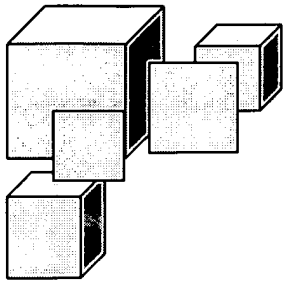
Software modules

- DeviceManager
  - The SDC Initiator side SCSI Transport layer used to interface to the FC / P-SCSI HBAs.
  - Implements an asynchronous RPC model to execute SCSI commands.
  - Exports only LU API (targets are not exposed). Internally maintains targets.
  - Maintains multiple "target paths" to available targets and can implement several strategies for choosing the path to take (Round Robin, Fail over, …).
  - Implements the SCSI Discovery logic
    - Target Ready, Report LUs, LU capacity.
  - Reports to EMM all new Remote LUs found.
  - Interacts with the FC / P-SCSI drivers to execute the commands



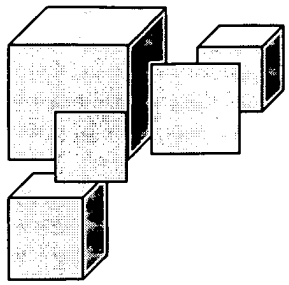
Software modules

- ISL
  - The SDC Target side SCSI Transport layer used to interface to the iSCSI stack.
  - Enable working with iSCSI from any CPU and from multiple tasks without paying the penalty of multiple messaging hops.
  - Supplies a callback model for receiving events from iSCSI and function calls for sending events to iSCSI.
  - Each operation with ISL is identified by a handle. This handle is an object that supplies all functionality that can be done for this operation. E.g. ISLInitiatorHandle.
  - API compels flow control mechanism (as appropriate for Target Side SCSI Transport).



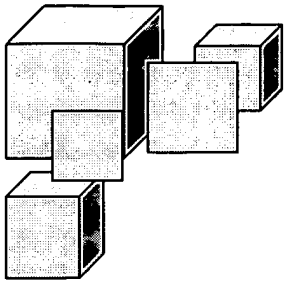
Software modules

- iSCSI Stack
  - Implements the iSCSI protocol SCSI Transport.
    - Currently supports rev08
    - The current draft number is rev17 (should be RFC in 3-4 months).
  - All execution is self contained i.e. interfacing with ISL separates via message queues the iSCSI low level flow. This insures non-blocking and small event length handling.
  - Build on top of the TCP/IP stack with enhanced event delivery mechanism.



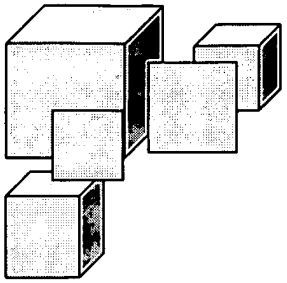
Software modules

- Networking
  - NetBSD TCP/IP stack ported by SANRAD to VxWorks for MIPS.
  - Having our own stack enabled us to:
    - Implement efficient event delivery (EED)
    - Achieve zero copy all the way
    - Support TCP extension for Gigabit (Window Scaling, TCP Timestamp Option, …).
  - Why NetBSD
    - Good documentation (TCP/IP Illustrated 2)
    - Open source ($$)
    - NetBSD is a living project so we will get updates (e.g. VLAN support, IPv6) and bug fixes to the stack.



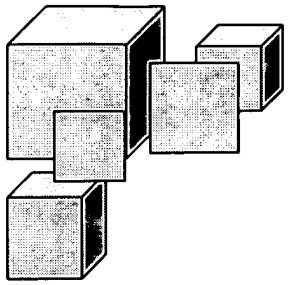
Software modules

- StorageDrivers
  - Implementation of the HBA drivers.
    - Currently VMIC FC, LSI P-SCSI and Agilent FC
  - Dynamically locate and configure HBAs detected on PCI over LDT.
  - HBA drivers implemented using SGLs (Scatter Gather List) for implementing zero copy.
  - Implements target discovery functionality for each protocol, notifying on new target info and lost target info to the DeviceManager.



Software modules

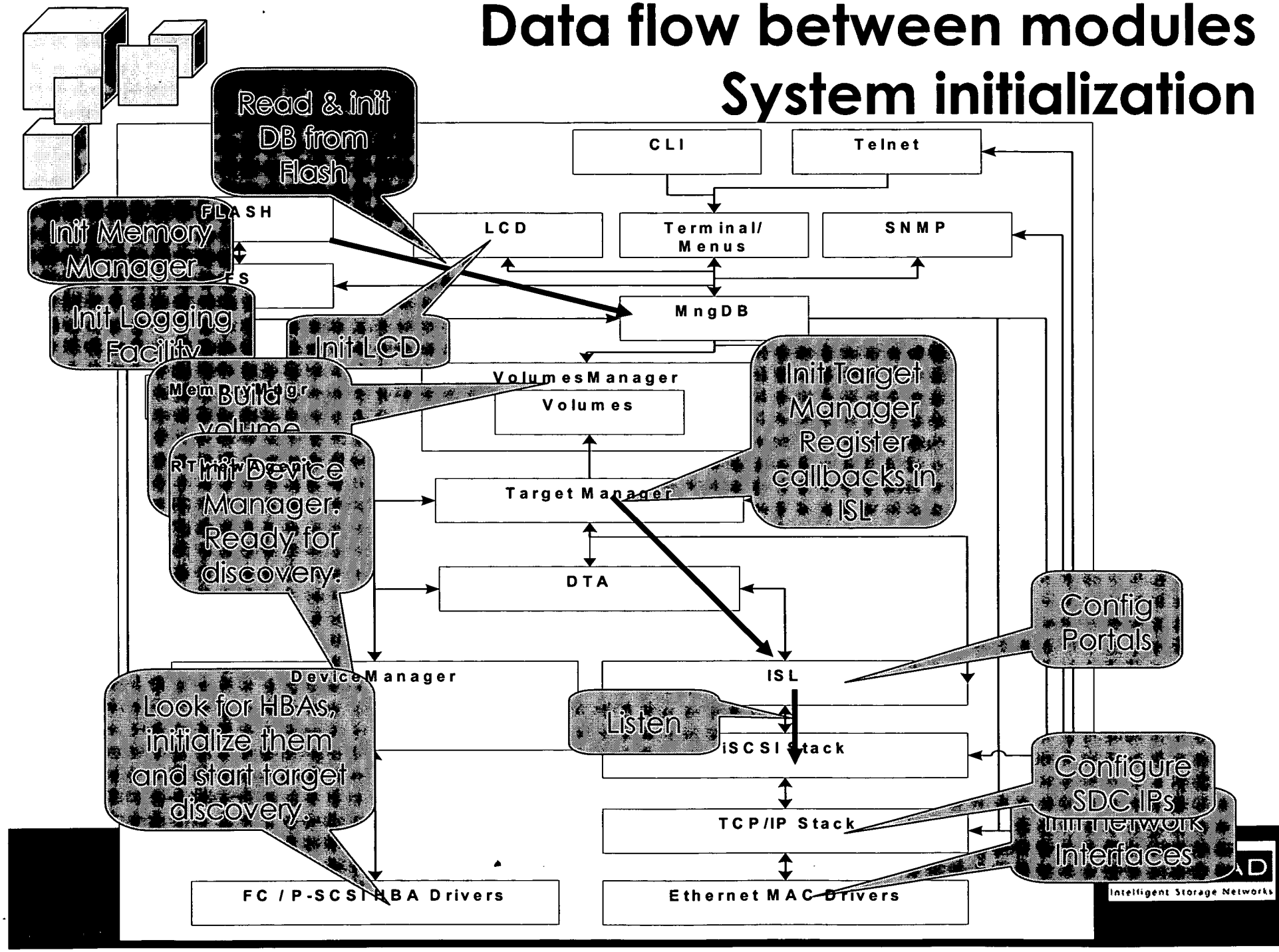
- MemoryMngr
  - Enable dynamic allocation and freeing of constant size objects (as oppose to malloc).
  - Dynamically shift system memory to the any consumer I.e. number of elements of each type are not constant.
  - Two main management types:
    - Local CPU – objects must be allocated and freed on the same CPU
    - Shared – objects can be allocated and freed on different CPUs.
  - Enables "wait for memory" type of allocation.
    - In the case of memory shortage, a task will be blocked until an element of the requested type can be allocated.
    - Should be used for non-critical missions. Examples might be: target discovery, CLI or user configuration in general.
  - Memory is statically divided between CPUs and shared part.



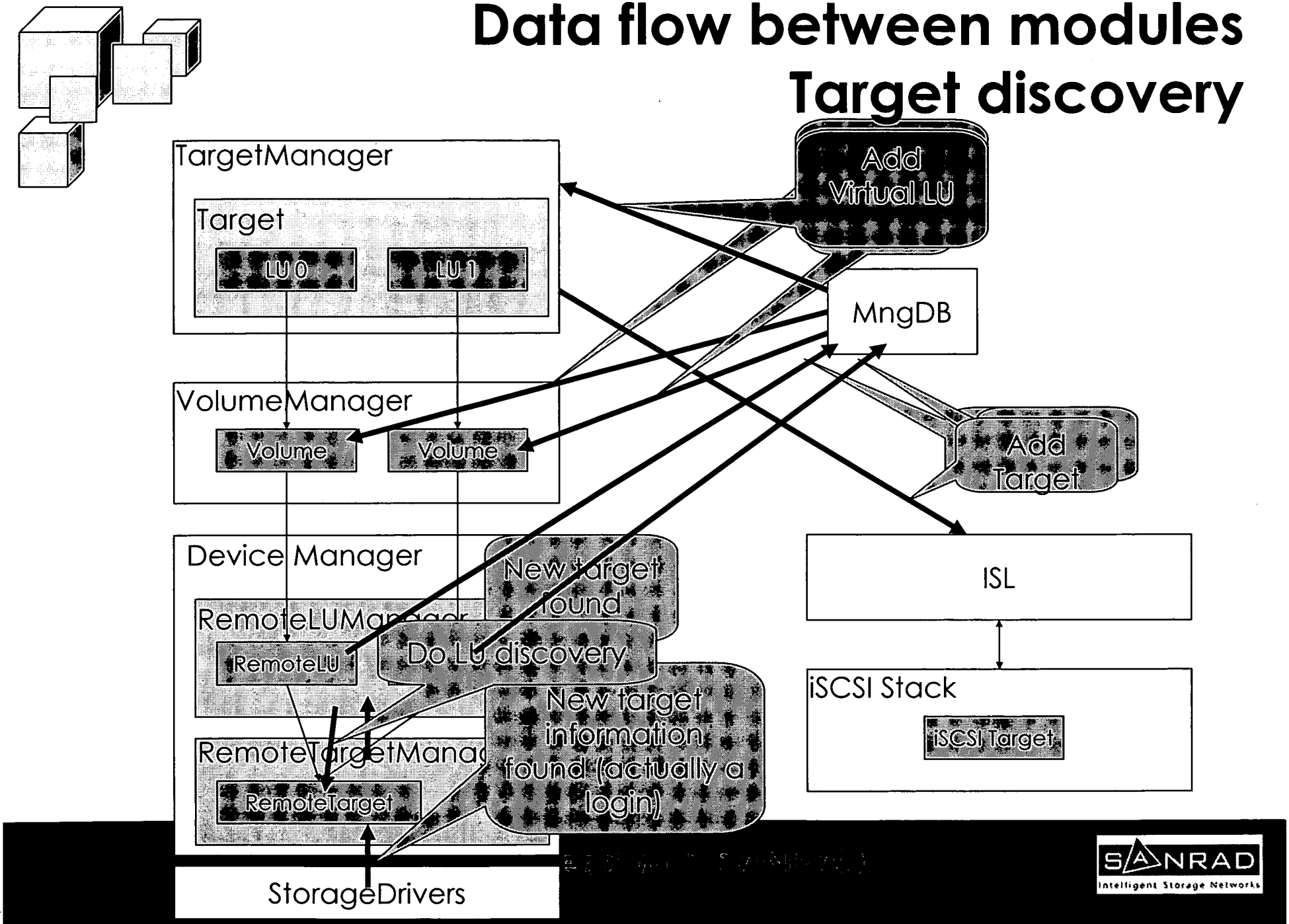
Software modules

- RTviewAgent - This part is mostly future
  - An agent that can send out debug information in a compact form.
  - Our code is filled with debug hooks.
  - Debug output can be directed to RS232 or use Rtview facilities.
  - Debug to RS232 is very expensive (string formatting and serial output).
  - Debug via RTviewAgent is more efficient.
  - Today can only be used when an external consumer application is connected to the agent.
  - In the future will be used offline in the field. Saving its condensed output to flash for SANRAD to download and view.

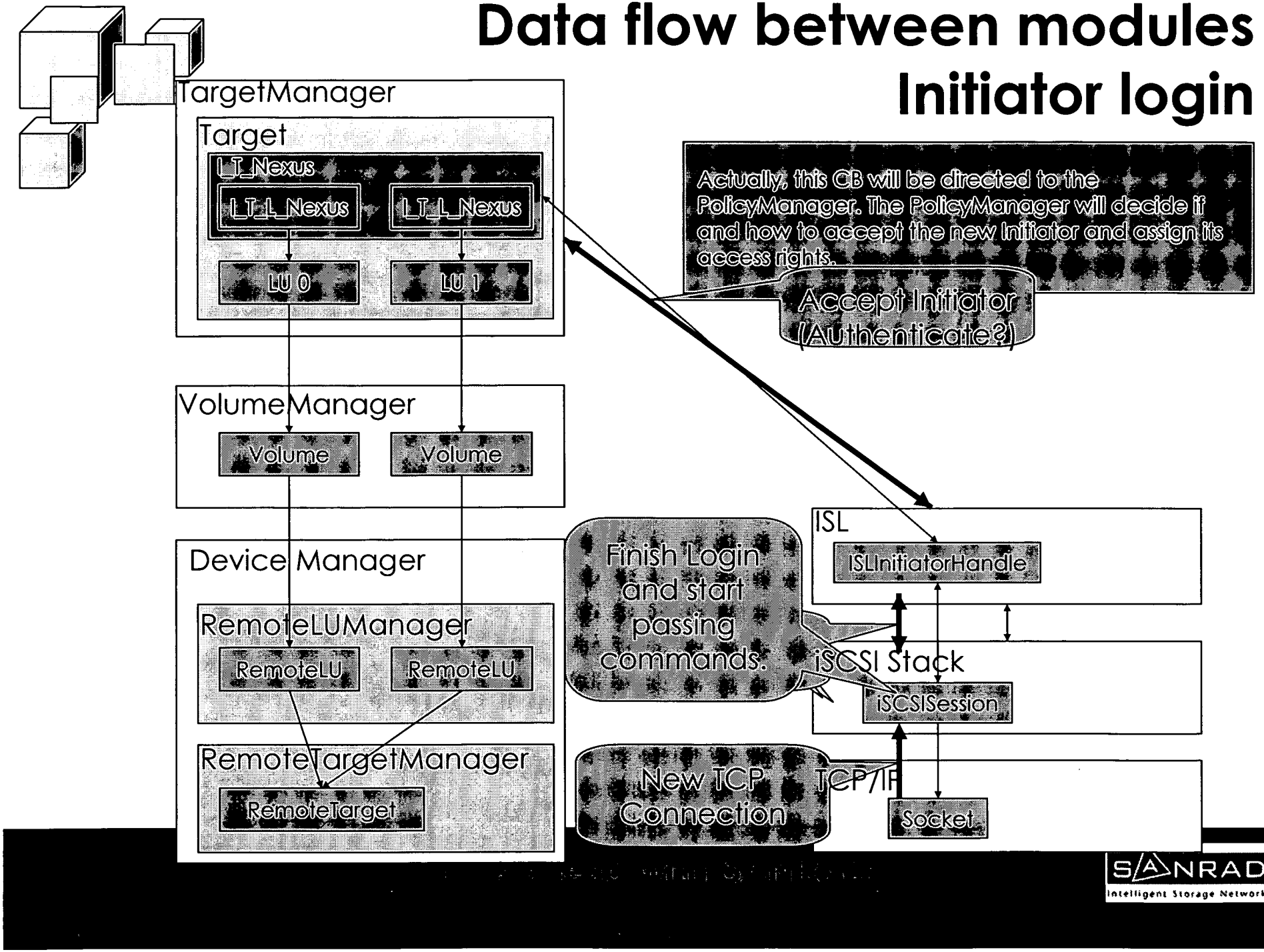
Data flow between modules
System initialization

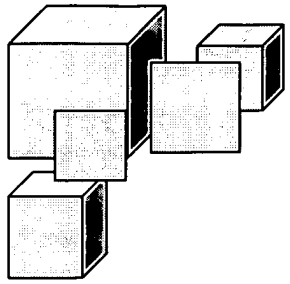


Data flow between modules
Target discovery



Data flow between modules
Initiator login

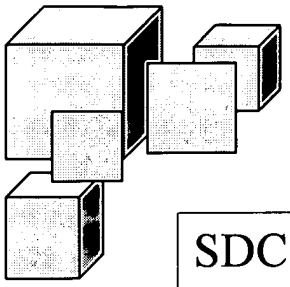


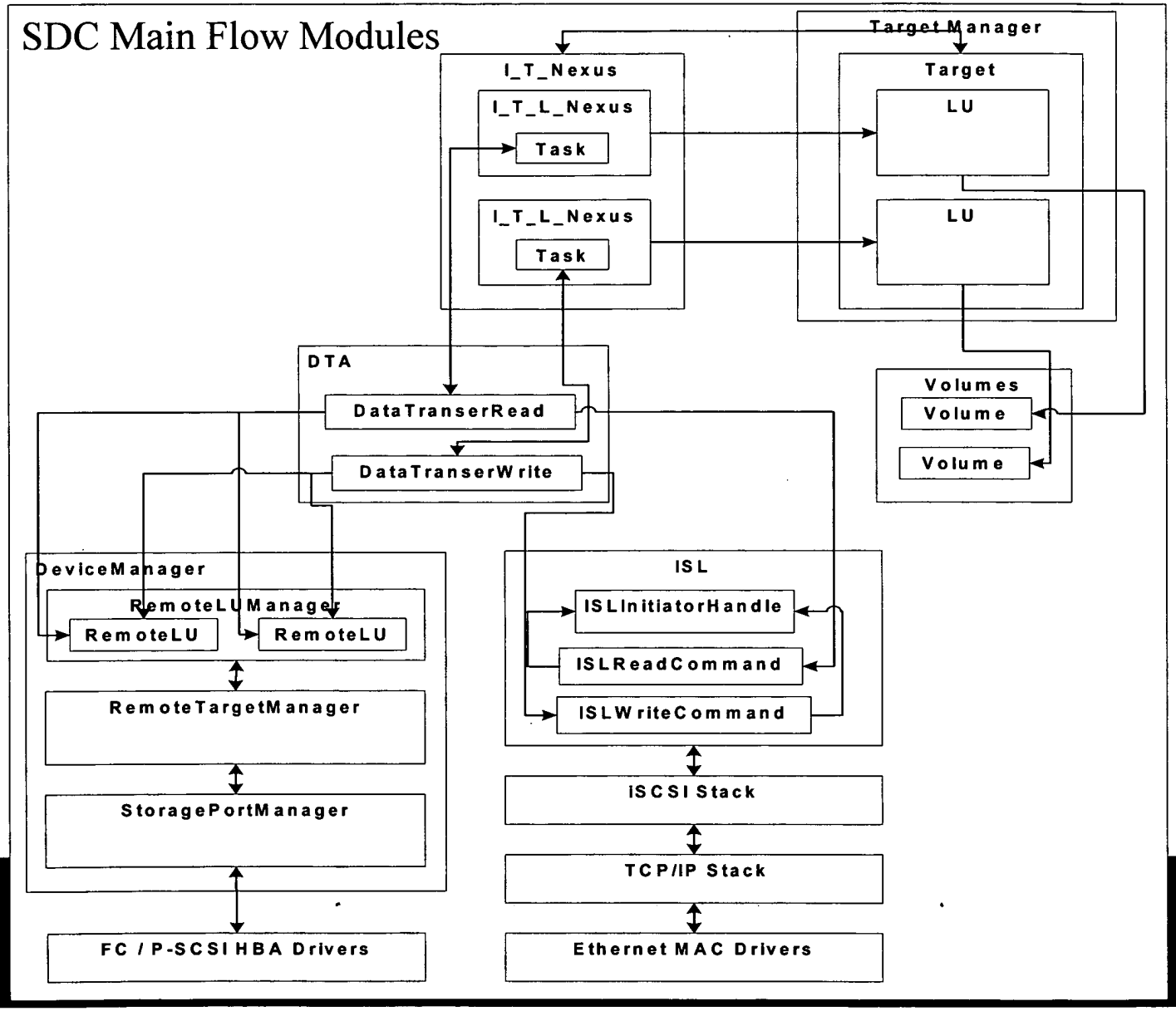


Data flow between modules Command Processing

- There are multiple commands processed in the system concurrently.
- Each command has its own state and continues its processing on events that are delivered to it.
- Thread-Based Concurrency as oppose to Event-Driven Concurrency.

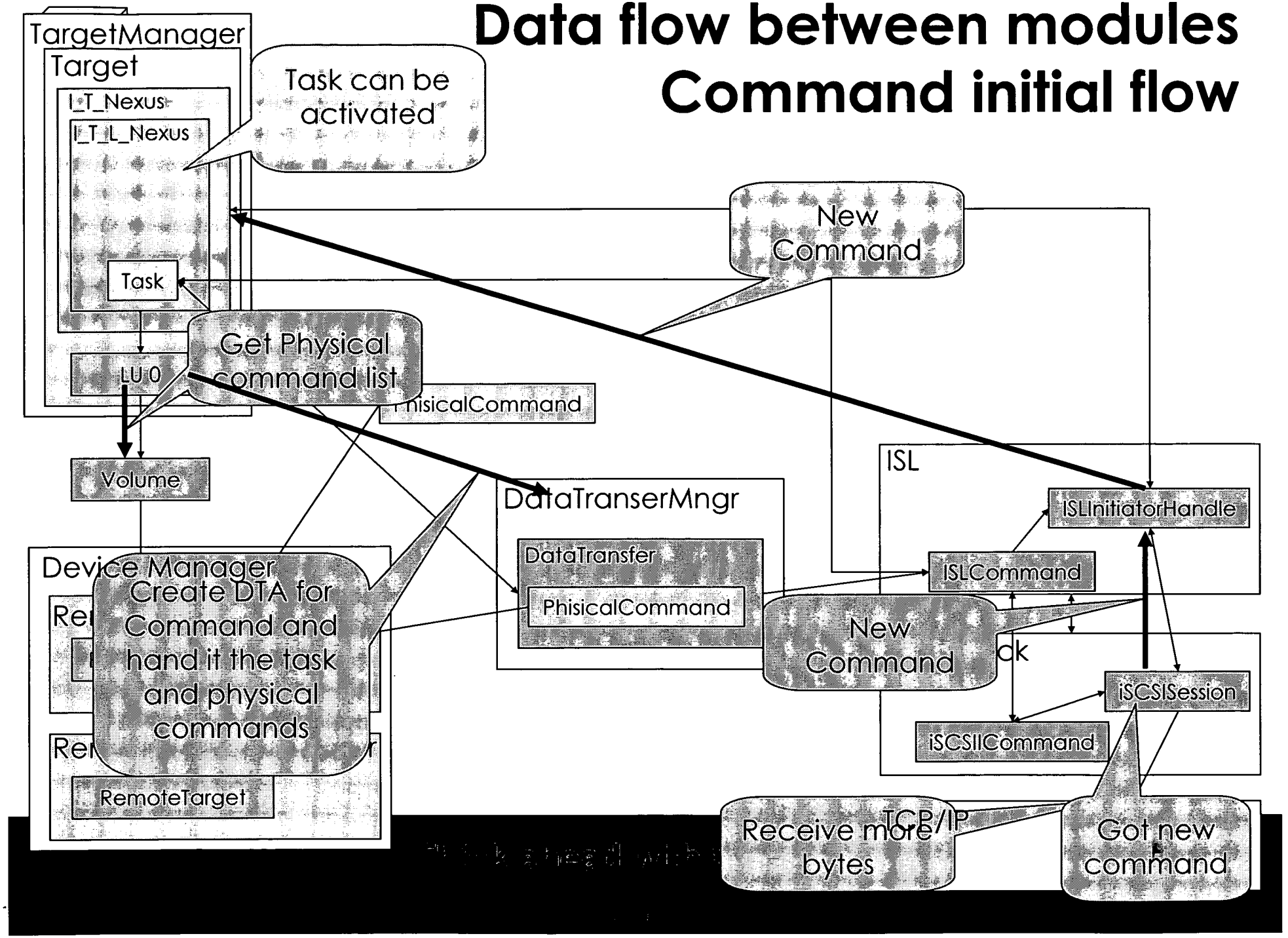
Data flow between modules Multiple Commands



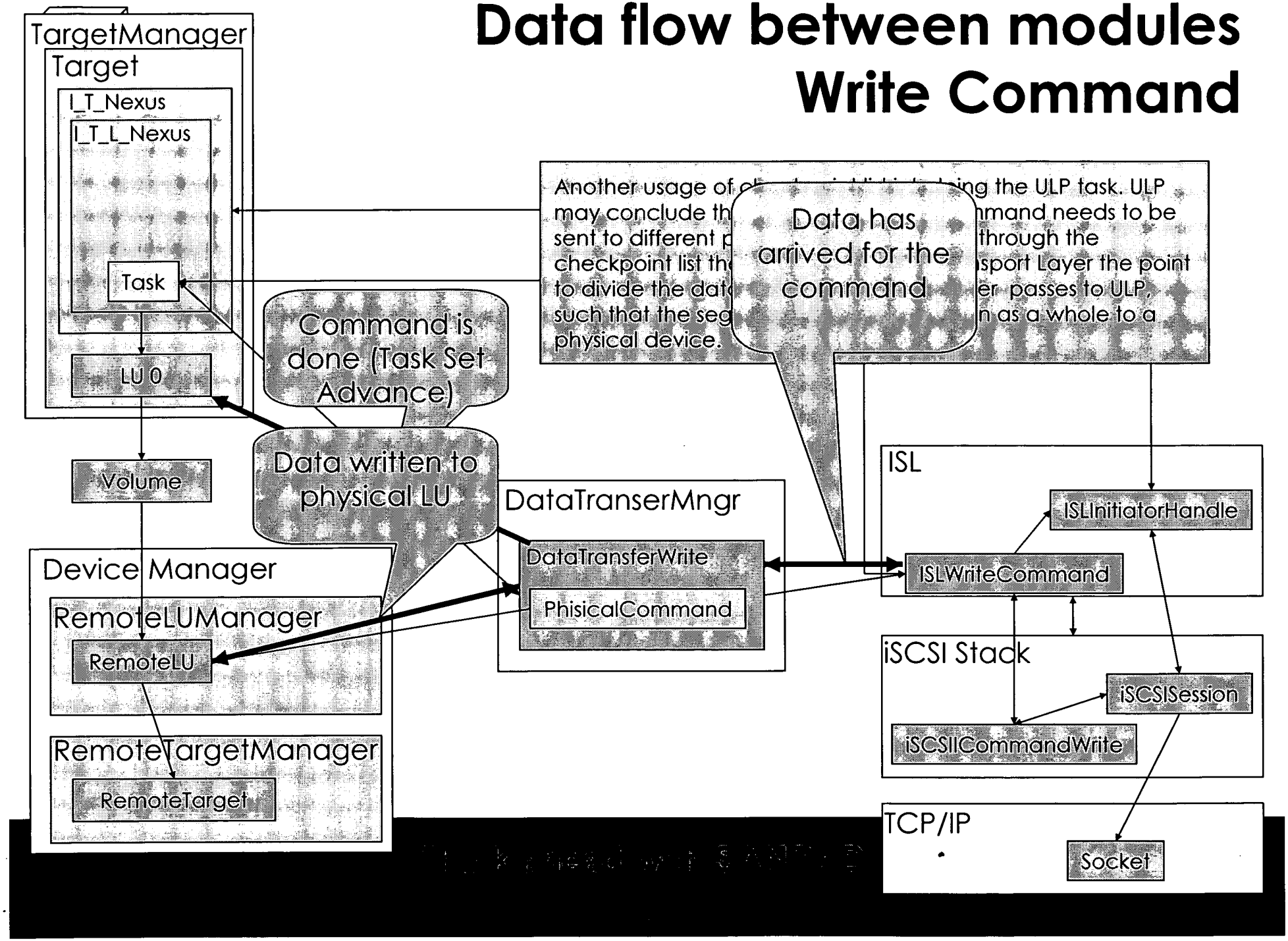


SANRAD
Intelligent Storage Networks

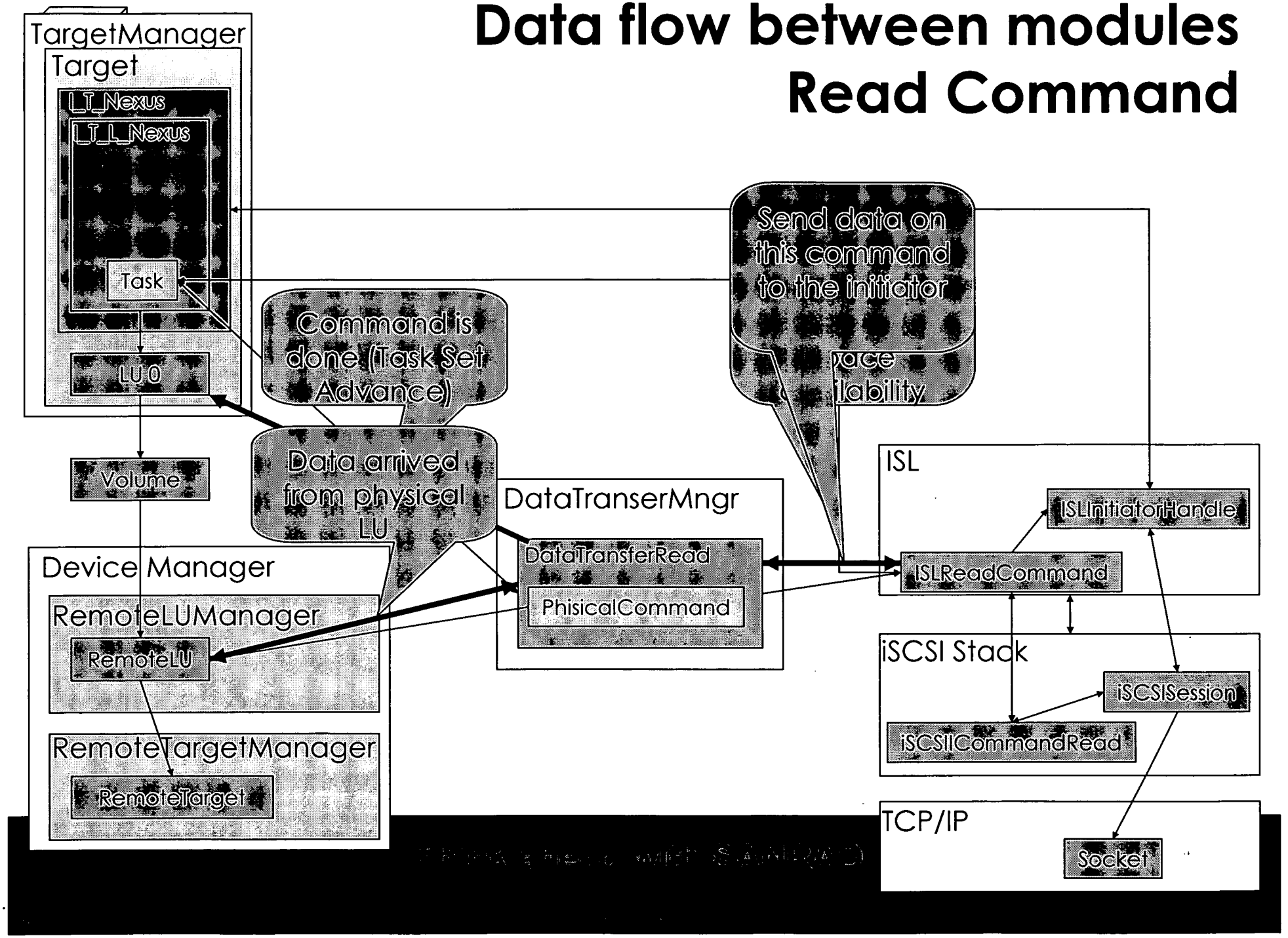
Data flow between modules
Command initial flow

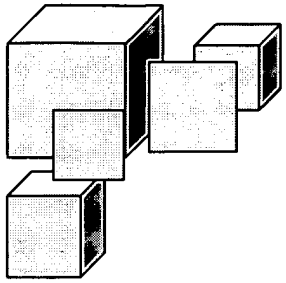


Data flow between modules
Write Command



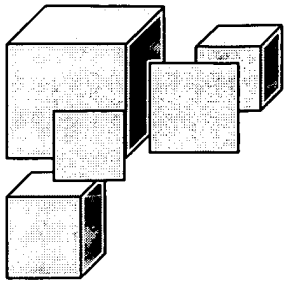
Data flow between modules
Read Command





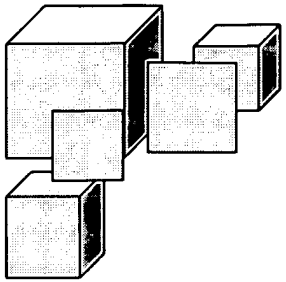
System Flow Control

- In SCSI the controlling figure is the Target.
- The SDC also implements flow control on the Target side.
- Flow Control is Implemented by:
  - using buffer limits per command.
    - Each command maintains a logical limit on the amount of memory that it will consume.
    - When this limit is reached, no more data will be added to this command until some of it is evicted from the SDC.
  - limiting the number of concurrent commands (per session).



System Flow Control (Cont)

- Write commands flow is controlled by using the RTT SCSI Transport mechanism.
- Read commands flow control is controlled by the output TCP connection that the command belongs to.
  - Transport types other than iSCSI will control it by other methods (HW-SW queues).
- iSCSI Flow Control
  - On receive iSCSI stack mandates flow control by using R2T. R2T computation is done by looking at the space left in the receive buffer.
  - TCP/IP flow control on receive in not used i.e. the connection is drained each time it has data. This is done since commands are multiplexed in a connection and by using TCP/IP flow control iSCSI might block an urgent Task Management command.
  - When sending TCP/IP does mandate flow control. So all commands on the blocked connection will fill their send buffer and eventually block.

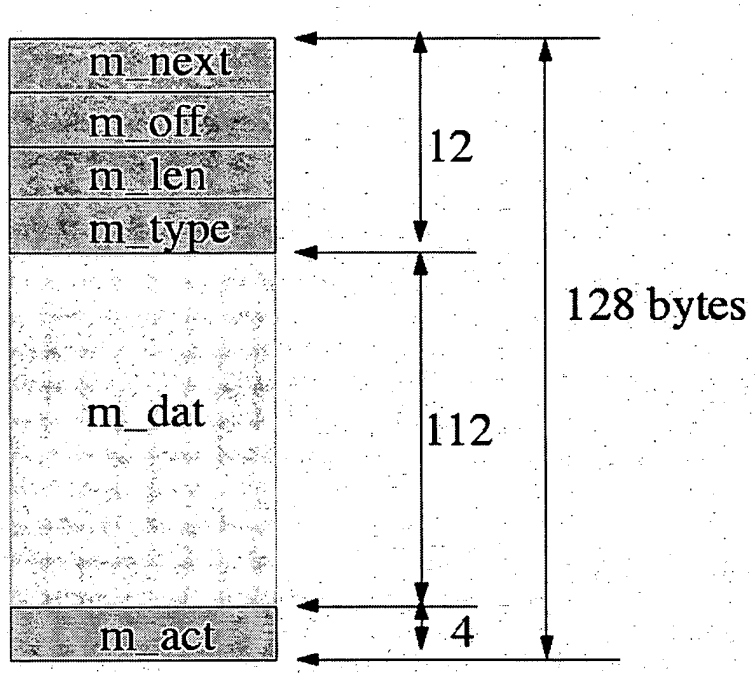


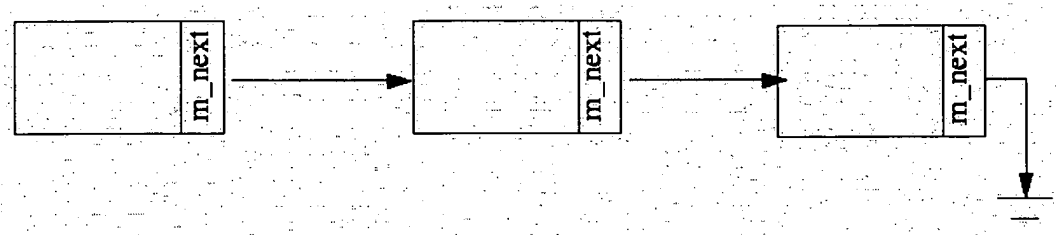
System Data Buffers

- The whole system passes around the SCSI data in buffers called SGL.
- An SGL stands for Scatter Gather List and is composed of multiple segments tied together in a linked list.
- An SGL represents a logical contiguous buffer.
- SGL is a C++ wrapper for MBUF (Memory Buffer).

System Data Buffers (Cont)

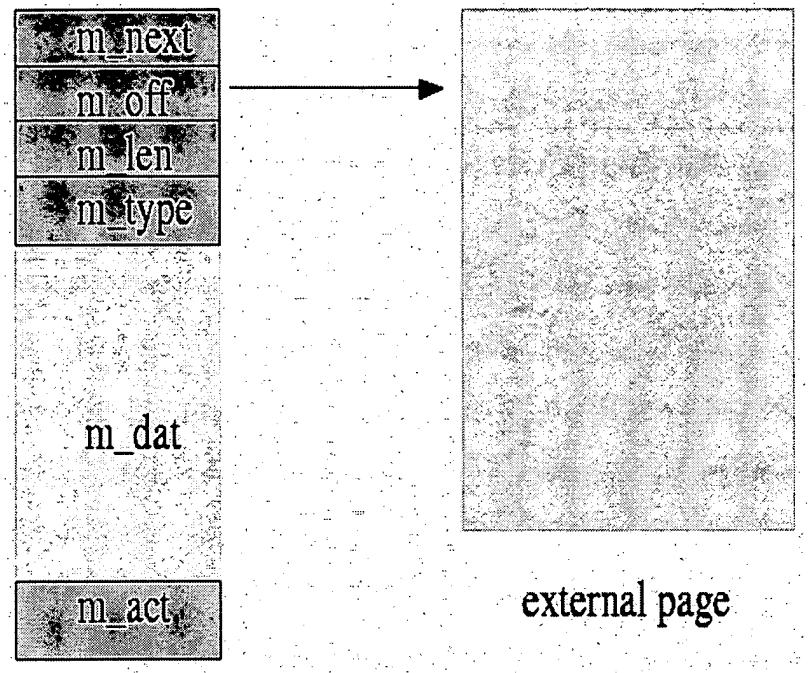
- This is the MBUF header.
- MBUFs are linked together to form an MBUF chain via their m_next field. This is actually an SGL.

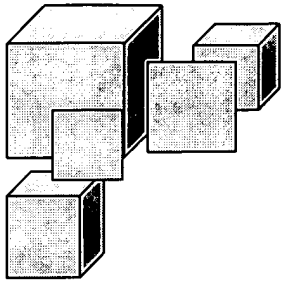




System Data Buffers (Cont)

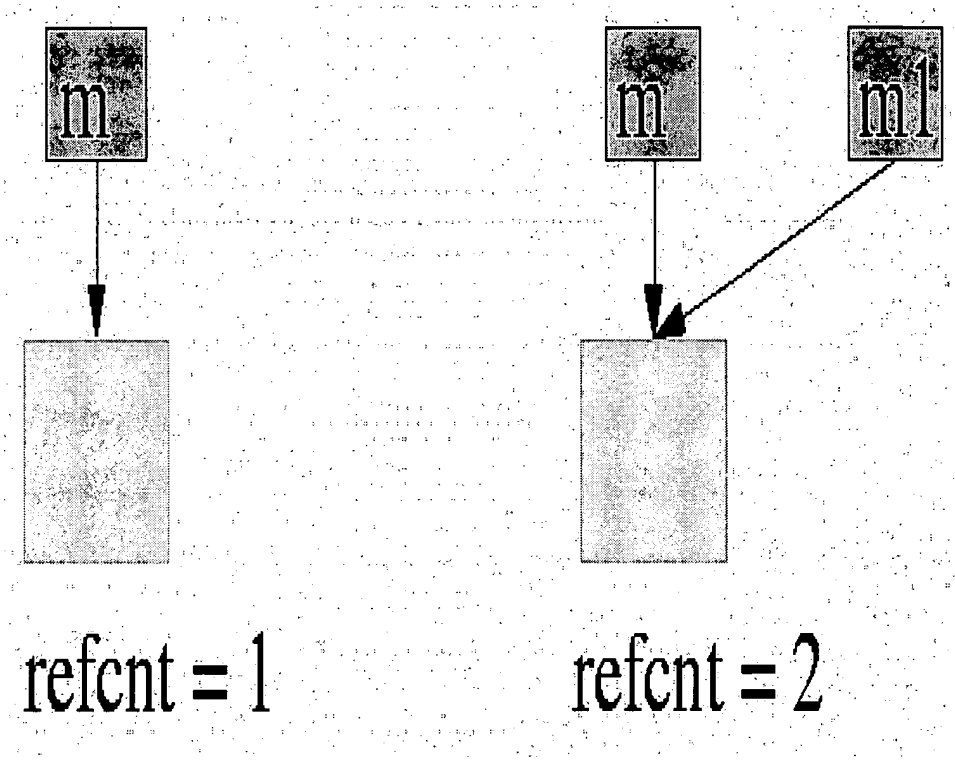
- When the MBUF segment is too small (it is only 100 bytes long). A CLUSTER (or an external buffer) is added.
- This way the MBUF is simply the control header and the data is kept in the CLUSTER.



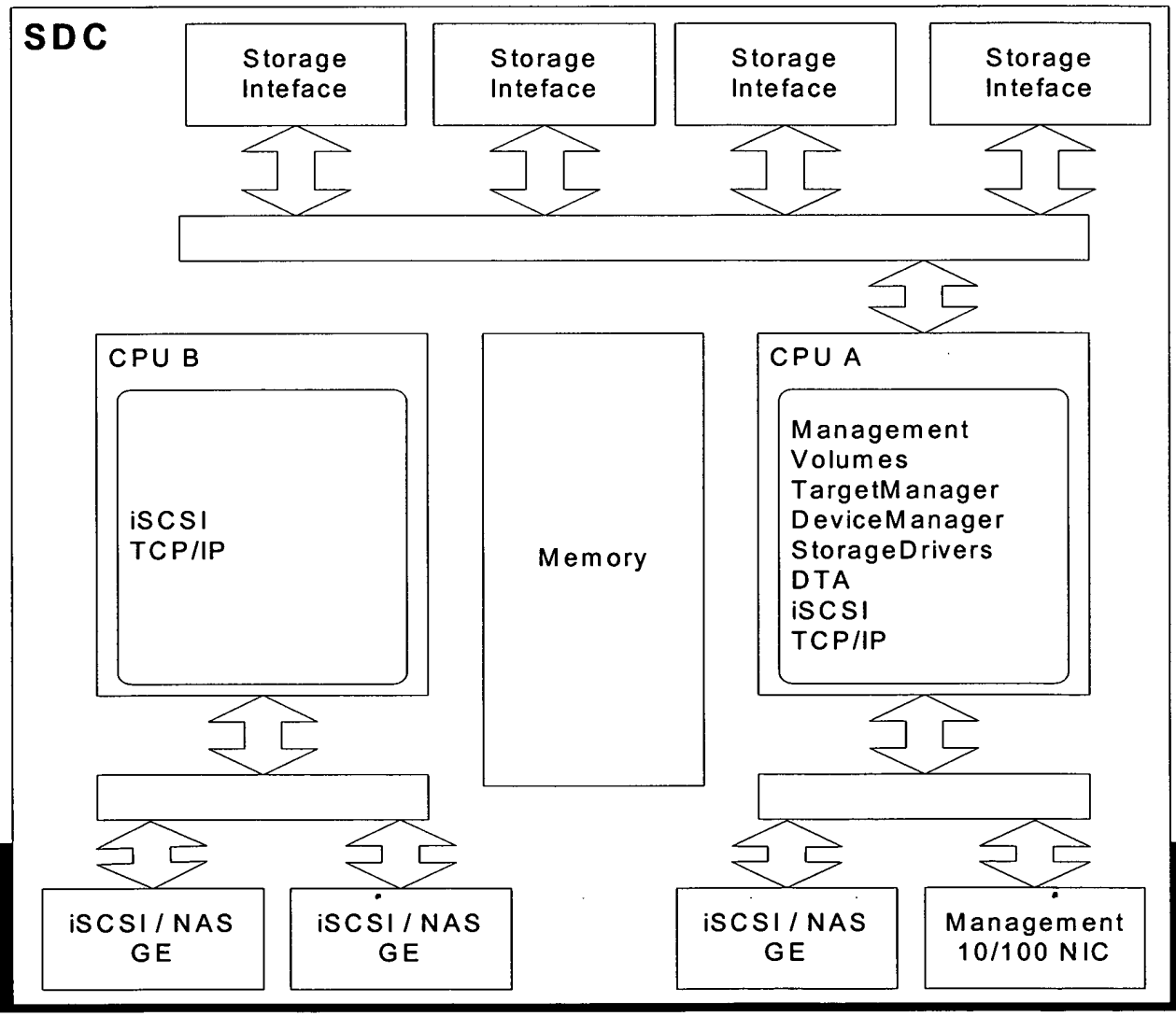


System Data Buffers (Cont)

- CLUSTERs can be referenced by multiple MBUFs.
- In this way we save copying the actual data but just add a reference to it.
- Copying a CLUSTER MBUF is done by:
  - Allocating a new MBUF.
  - Pointing to the CLUSTER.
  - Increment the reference count of the CLUSTER.

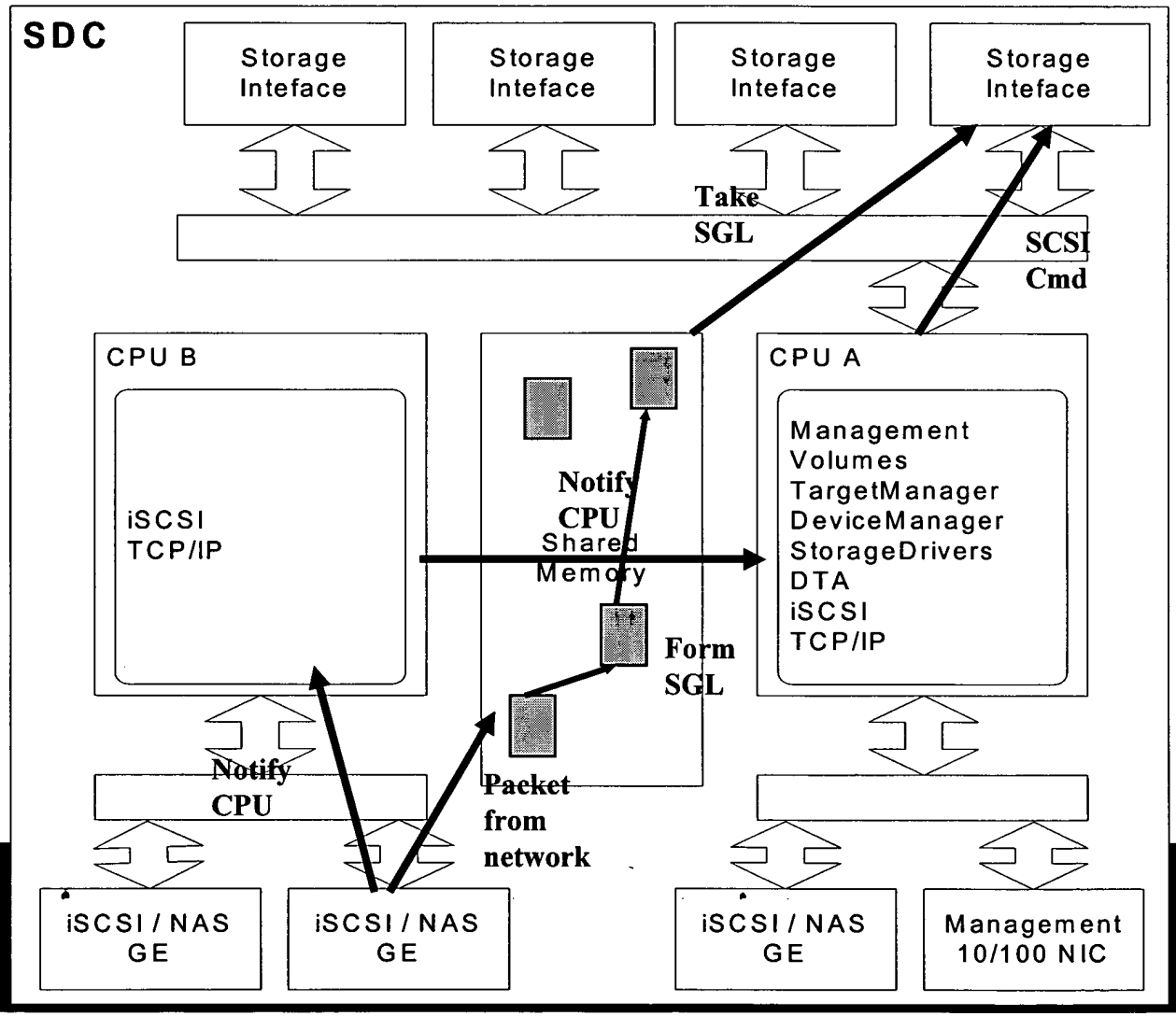


Software layout on hardware and explicit data flow



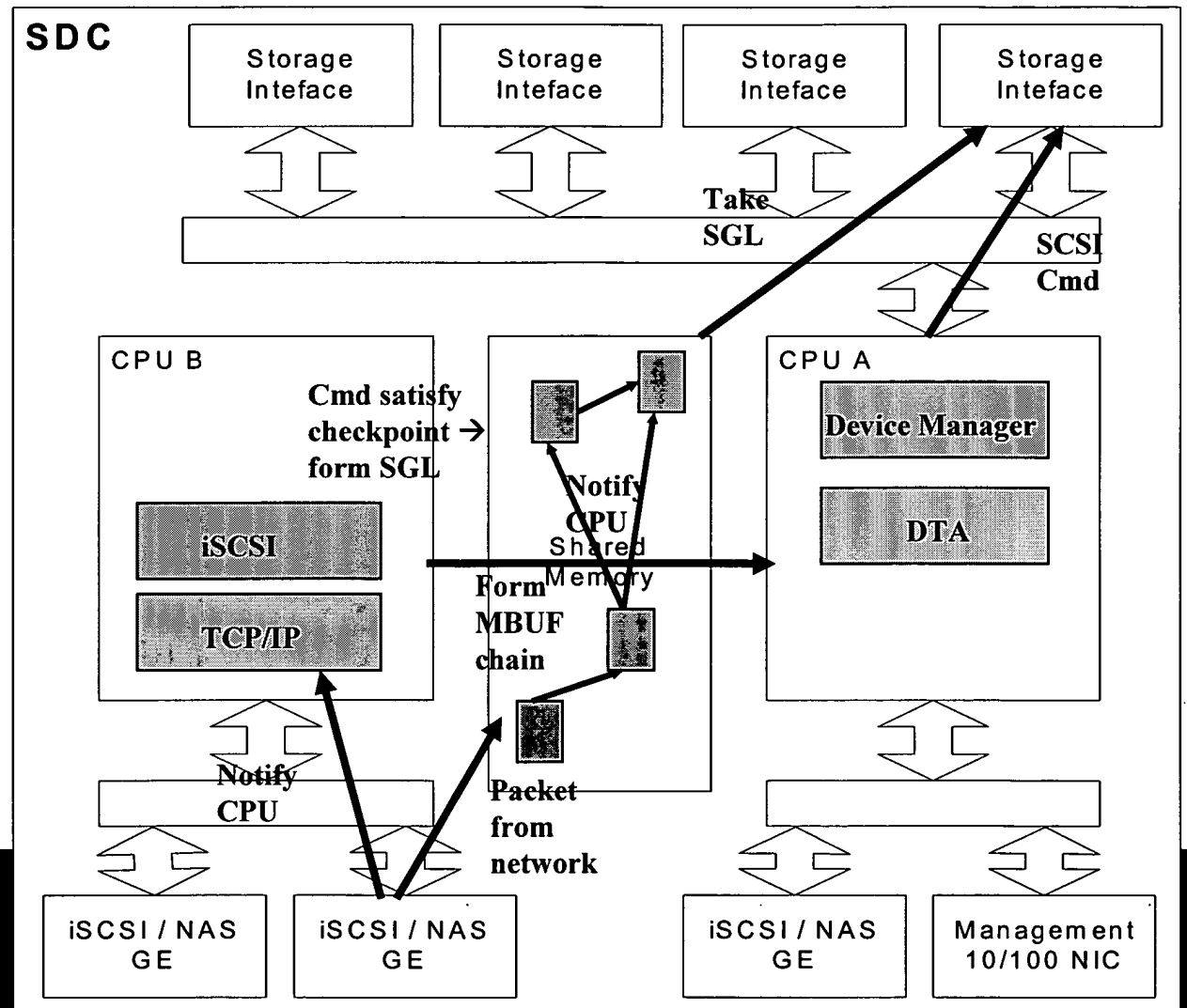
SANRAD
Intelligent Storage Networks

Software layout on hardware and explicit data flow (Cont)



SANRAD
Intelligent Storage Networks

Software layout on hardware and explicit data flow (Cont) - write



SANRAD
Intelligent Storage Networks

EXHIBIT 4

Go to Google Desktop

Web Images Video News Maps **Desktop** more »

Ronny Sayag Search Desktop Preferences Advanced Search

Cached messages

Message **5** of **5** in conversation

« Older | Newer » View Entire Thread (5) Reply | Reply to all | Forward | View in Outlook

RE: SAN-005

From: Michael Ben-Shimon <michael@enitiatives.biz>
To: Ronny Sayag <ronny@SANRAD.COM>
Date: Jun 03 2003 - 5:35am

Confirmed.

-----Original Message-----
From: Ronny Sayag [mailto:ronny@SANRAD.COM]
Sent: Monday, June 02, 2003 4:58 PM
To: Michael Ben-Shimon
Subject: RE: SAN-005

Hi Michael ,

15/6 , 11:00 AM is OK .

Thanks,
Ronny

-----Original Message-----
From: Michael Ben-Shimon [mailto:michael@enitiatives.biz]
Sent: Monday, June 02, 2003 5:46 AM
To: Ronny Sayag
Subject: RE: SAN-005

Hi Ronny,

It would be a good idea to bring your team to the meeting. This invention targeted to encapsulate the virtualization concepts introduce in your product and specially the virtualization within the data path. I suggest to move the meeting to Sunday 15/6 at 11am. Please let me know if it works for you.

Thanks,

Michael
-----Original Message-----
From: Ronny Sayag [mailto:ronny@SANRAD.COM]
Sent: Friday, May 30, 2003 6:25 PM
To: Michael Ben-Shimon
Cc: Gadi Erlich
Subject: RE: SAN-005

Hi Michael ,

I will not be able to make it on Thu 12/June as I will be out of Sanrad most of the day for a seminar .

Please let me know the scope of the meeting as this patent might involve few eng. both from the iSCSI and Virtualization teams.
It might be good idea to have a meeting with all the involved in order to prevent multiple meetings . It really depends on how deep you want to get into the technical issues .

Please let me know an alternative date convenient for you to have the meeting.

Thanks,
Ronny

-----Original Message-----
From: Michael Ben-Shimon [mailto:michael@enitiatives.biz]
Sent: Friday, May 30, 2003 4:55 AM
To: Ronny Sayag
Cc: Gadi Erlich
Subject: SAN-005

Hi Ronny,

I'd like to schedule a meeting to have an in-depth interview on the virtualization patent. I suggest to have the meeting on Thursday, June 12. Please let me know a suitable time to you.

Regards,

Michael Ben-Shimon
eNitiatives - New Business Architects Ltd.

Phone: +972-9-8890502
Mobile: +972-53-598686

This message may contain confidential information intended for the use of addressee only.

« Older | Newer » View Entire Thread (5) Reply | Reply to all | Forward | View in Outlook

Ronny Sayag Search

Google Desktop Home - Browse Timeline - Index Status - Privacy - About - ©2007 Google